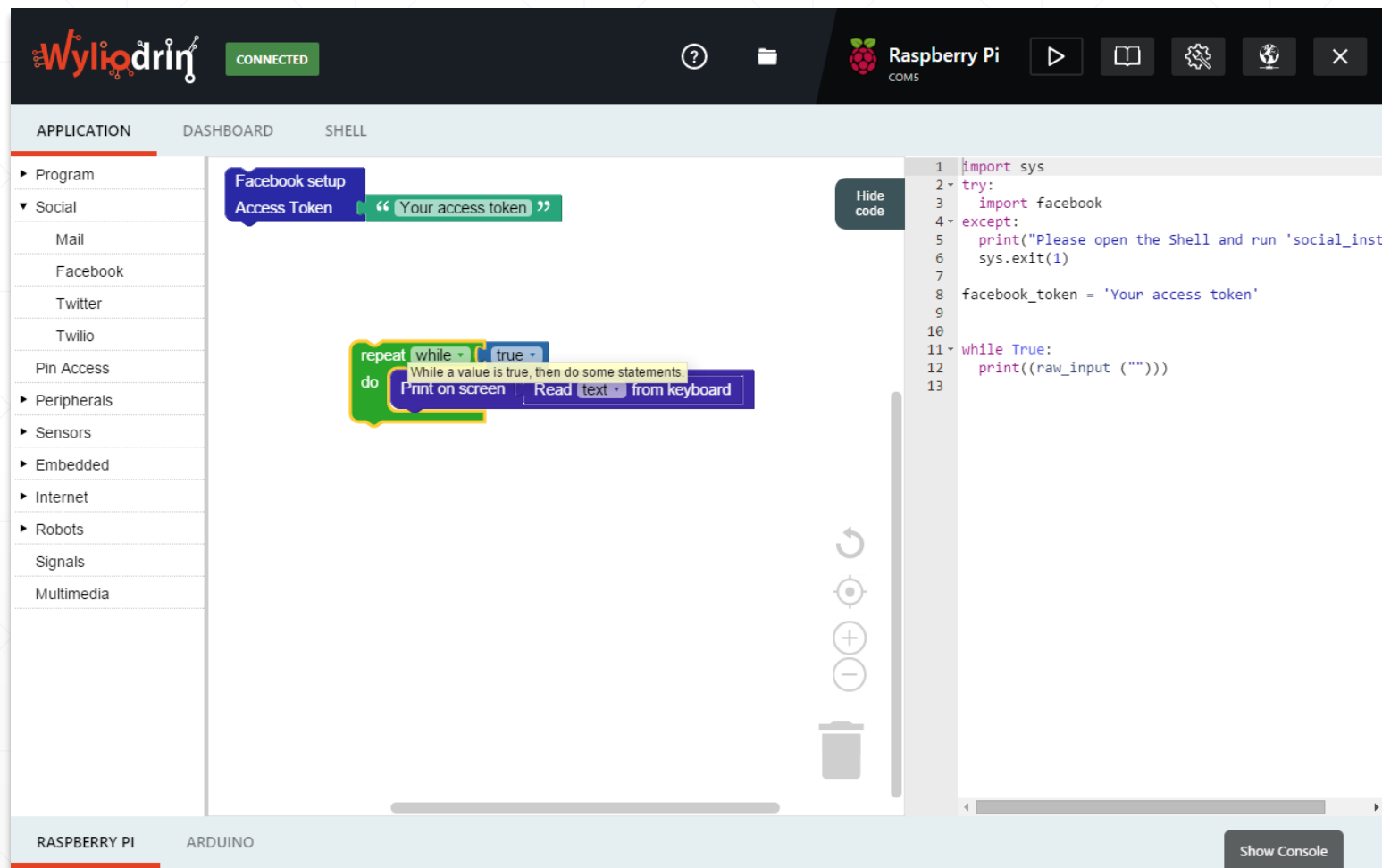


Wylidrin STUDIO

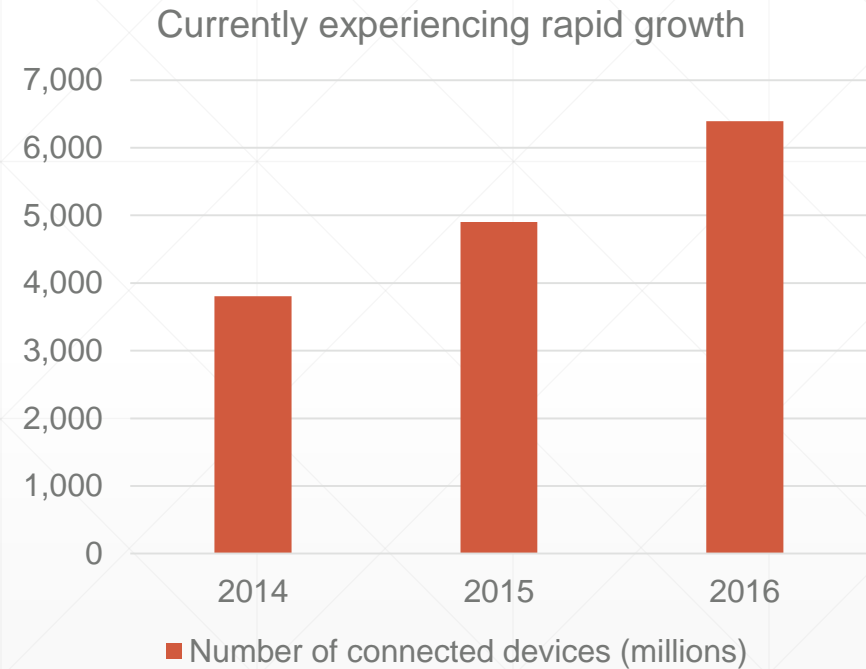
An Open Source Tool for IoT Development

What is the product



The technology: Hardware

- Before Raspberry Pi
 - expensive embedded devices
 - few devices
- Raspberry Pi changed the game



Our journey: The vision

- Goal:
 - A new approach towards engineering
 - IoT accessible to everyone

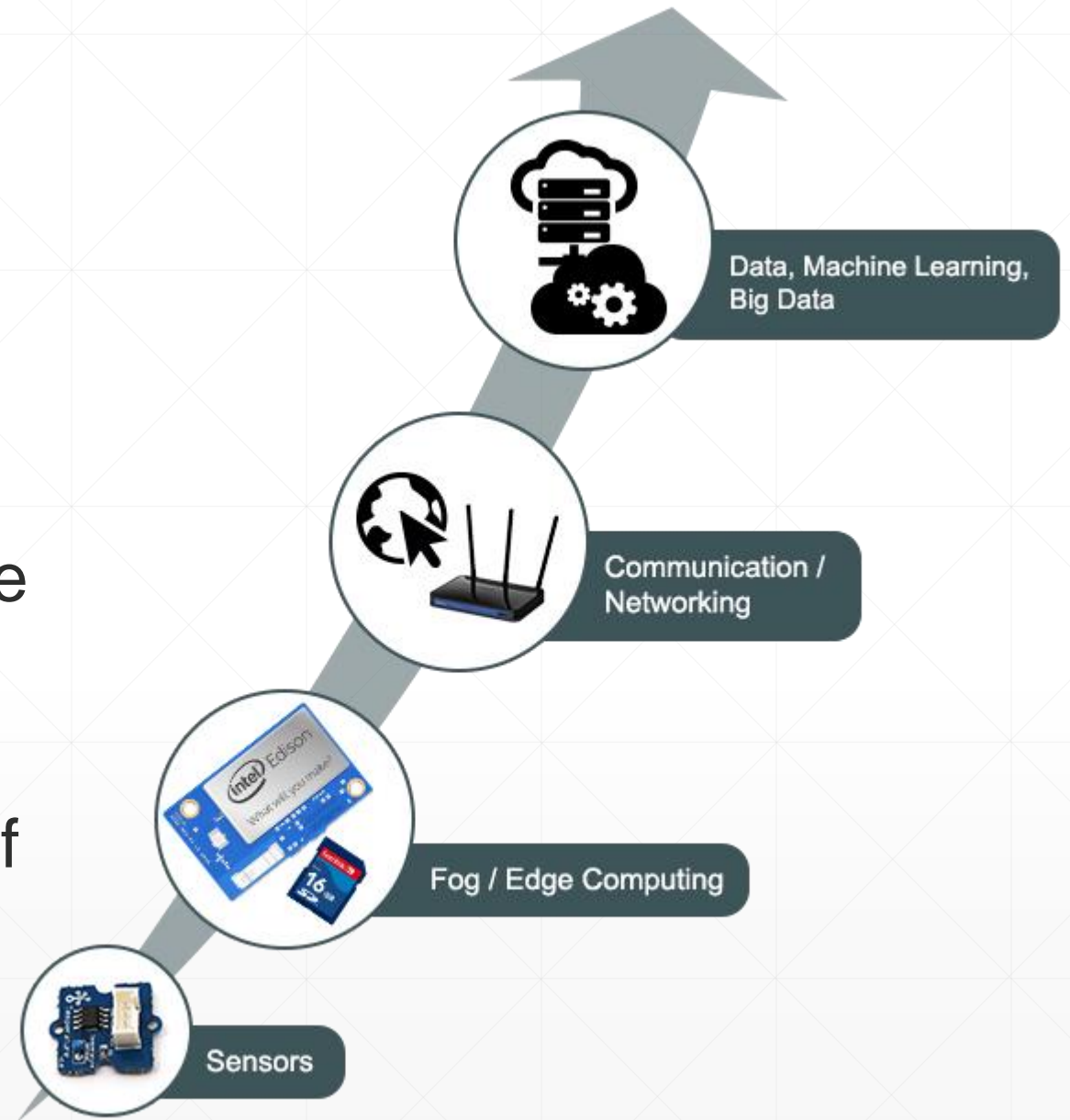


Create, modify, tweak, customize current solutions to your needs and use cases

The IoT stack

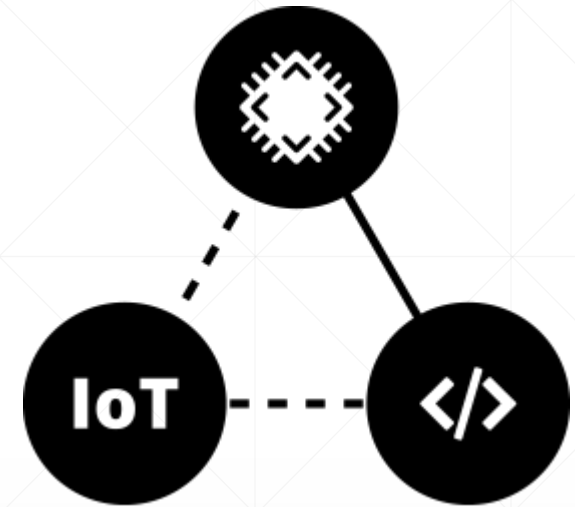
The problem

- Arduino (Uno) does well on Level 2 but does not follow the upper stack
- Raspberry Pi follows the full stack, but lacks the benefits of Arduino



Microcontrollers vs Embedded Boards

- Arduino Yun preferred to Raspberry Pi
- The fault
 - development tools
 - accessibility



Most of the projects are not IoT projects, they fall into electronics or programming

The solution

- Transfer the accessibility typical of Arduino to Raspberry Pi



Ease to use



Direct access



**High
productivity**



**Use from
anywhere**

Our tools for IoT : Wyliodrin

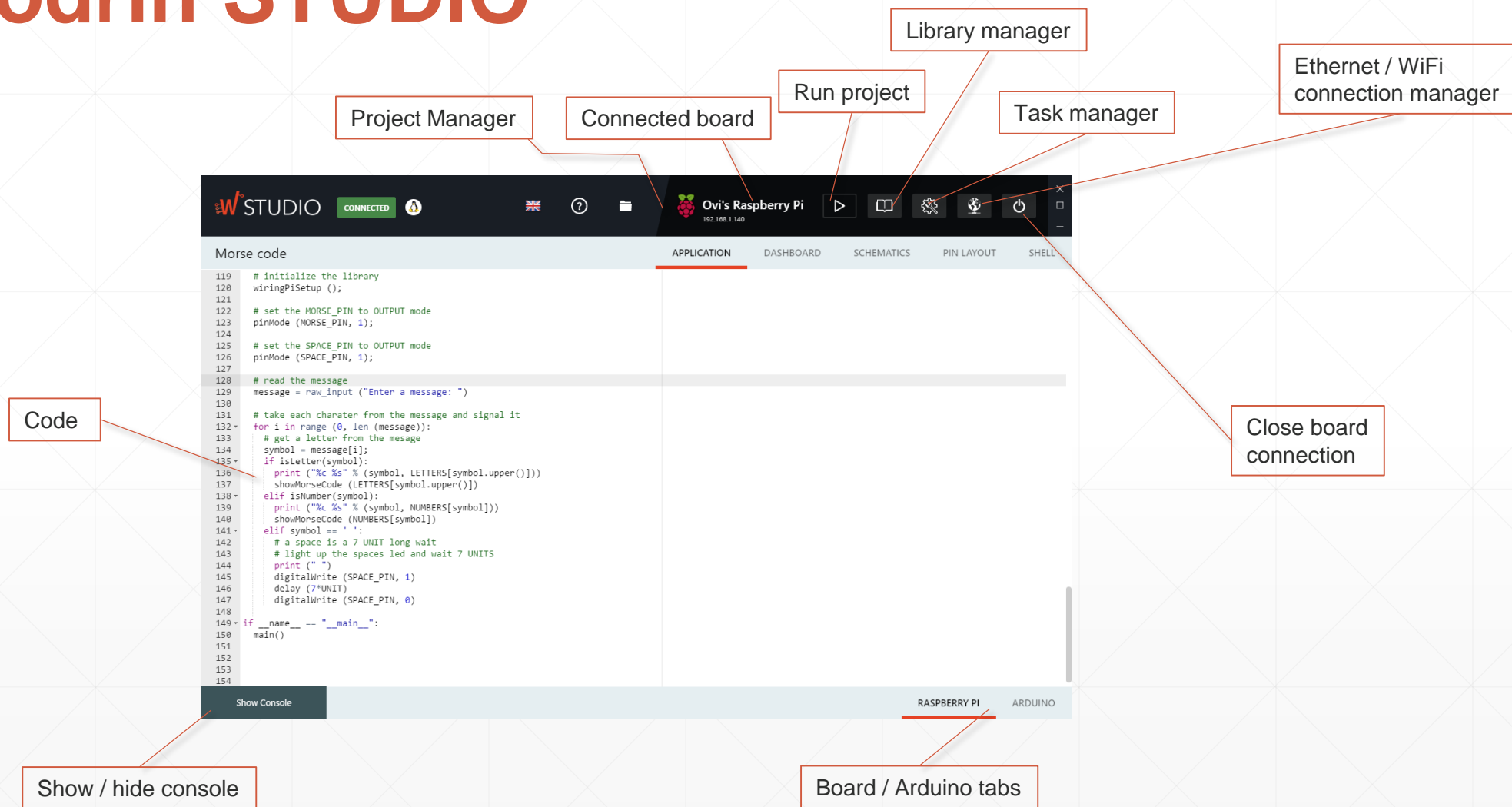
- Since 2013
- Fully Web-based
- Complex IDE
- Open Source components
- Free for basic use
- Supports various hardware: **Arduino Yun, Raspberry Pi, Intel® Galileo, Intel® Edison, UDOO, BeagleBone Black**



Wylidrin STUDIO

- Open Source
- Available for
 - Arduino Yun
 - UDOO Neo
 - Raspberry Pi
 - BeagleBone Black
- Works locally

Wylidrin STUDIO



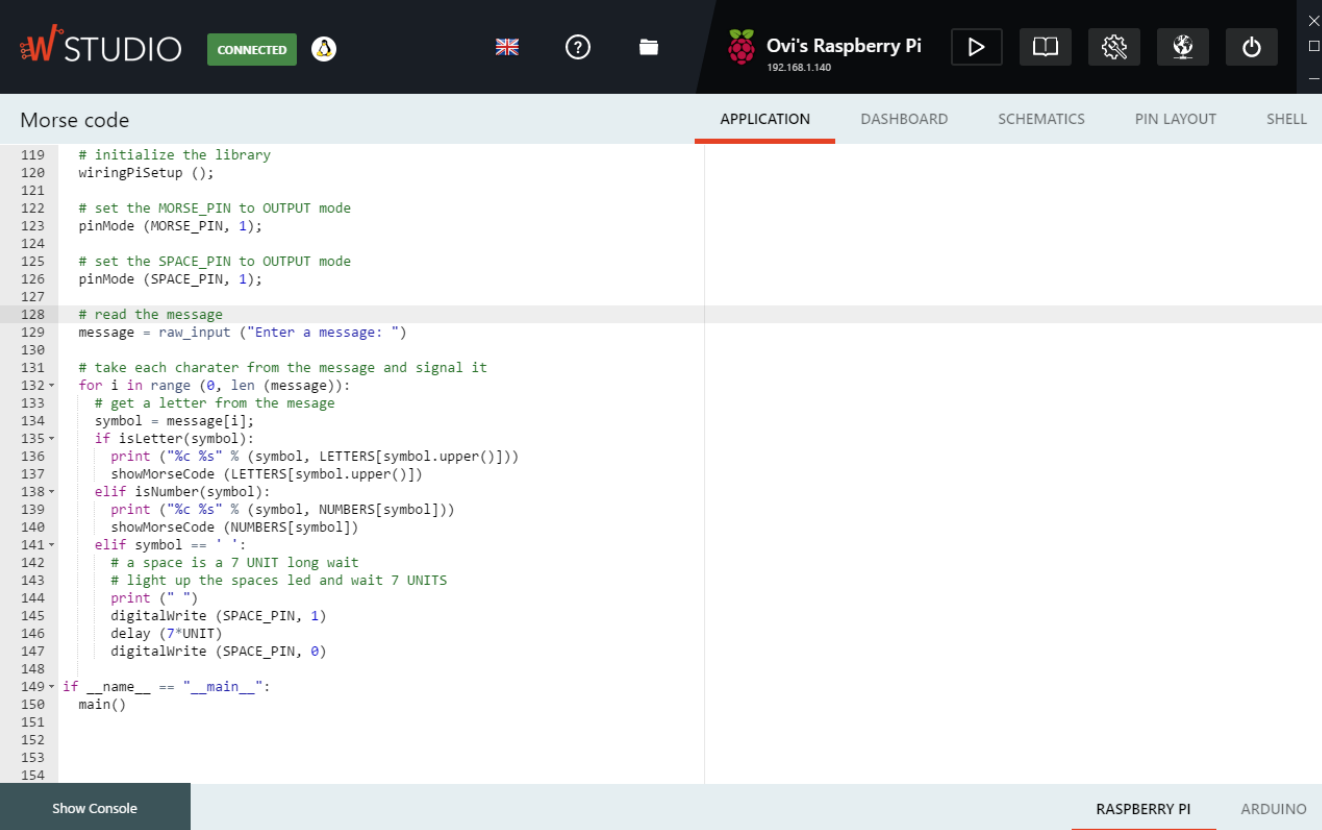
Board Connection

- Direct connection
 - Serial communication
 - Remote connection
 - Uses mDNS to discover devices in the same network
-

Programming

Professional code editor

Advanced features such as autocomplete



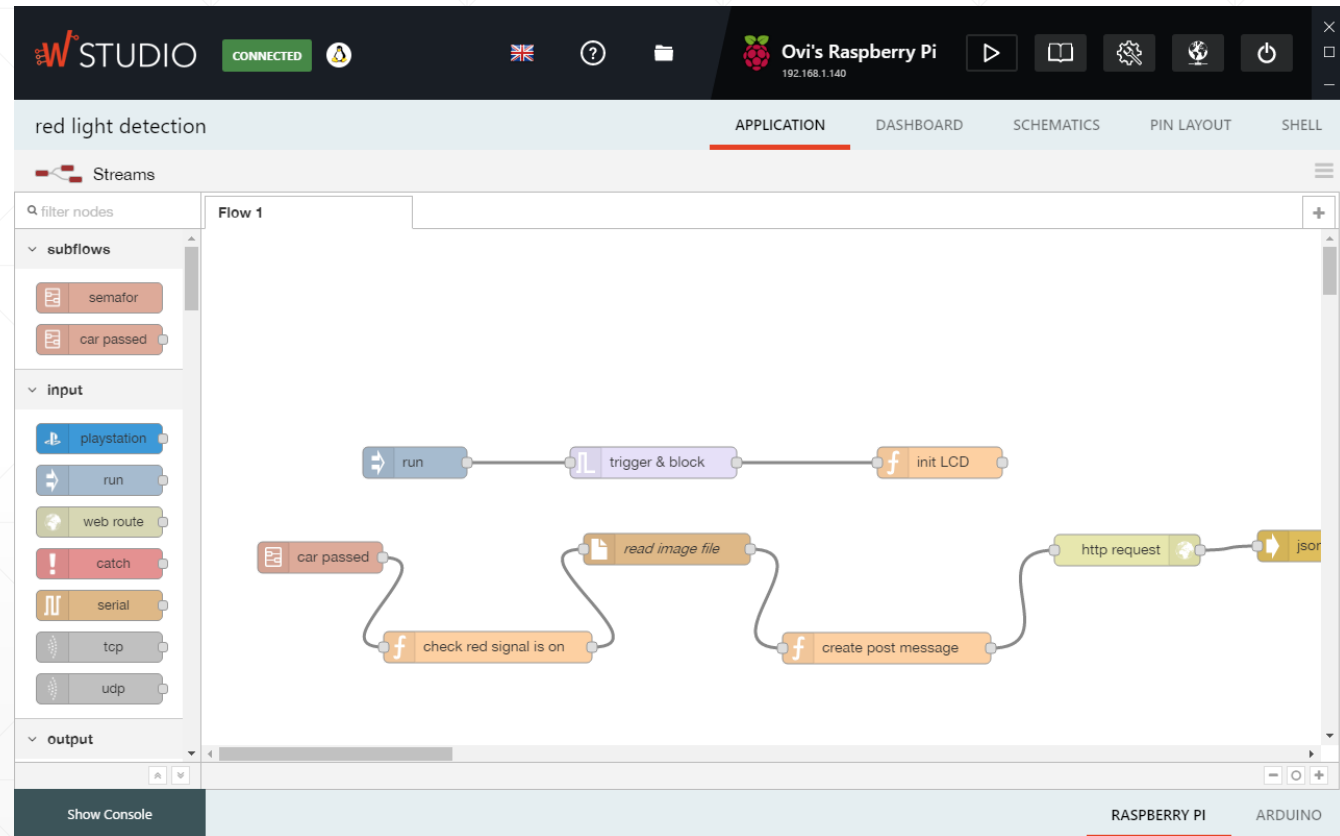
The screenshot displays the W Studio IDE interface. The top bar includes the 'W STUDIO' logo, a 'CONNECTED' status indicator, and various system icons. The main workspace is titled 'Morse code' and contains a Python script. The script initializes the wiringPi library, sets up Morse and Space pins, and implements a function to convert input characters into Morse code using a predefined dictionary. The interface also features a sidebar with navigation options like 'APPLICATION', 'DASHBOARD', 'SCHEMATICS', 'PIN LAYOUT', and 'SHELL'. At the bottom, there is a 'Show Console' button and a tab for 'RASPBERRY PI'.

```
119 # initialize the library
120 wiringPiSetup ();
121
122 # set the MORSE_PIN to OUTPUT mode
123 pinMode (MORSE_PIN, 1);
124
125 # set the SPACE_PIN to OUTPUT mode
126 pinMode (SPACE_PIN, 1);
127
128 # read the message
129 message = raw_input ("Enter a message: ")
130
131 # take each charater from the message and signal it
132 for i in range (0, len (message)):
133     # get a letter from the message
134     symbol = message[i];
135     if isLetter(symbol):
136         print ("%c %s" % (symbol, LETTERS[symbol.upper()]))
137         showMorseCode (LETTERS[symbol.upper()])
138     elif isNumber(symbol):
139         print ("%c %s" % (symbol, NUMBERS[symbol]))
140         showMorseCode (NUMBERS[symbol])
141     elif symbol == ' ':
142         # a space is a 7 UNIT long wait
143         # light up the spaces led and wait 7 UNITS
144         print (" ")
145         digitalWrite (SPACE_PIN, 1)
146         delay (7*UNIT)
147         digitalWrite (SPACE_PIN, 0)
148
149 if __name__ == "__main__":
150     main()
151
152
153
154
```

Streams

Data flow programming

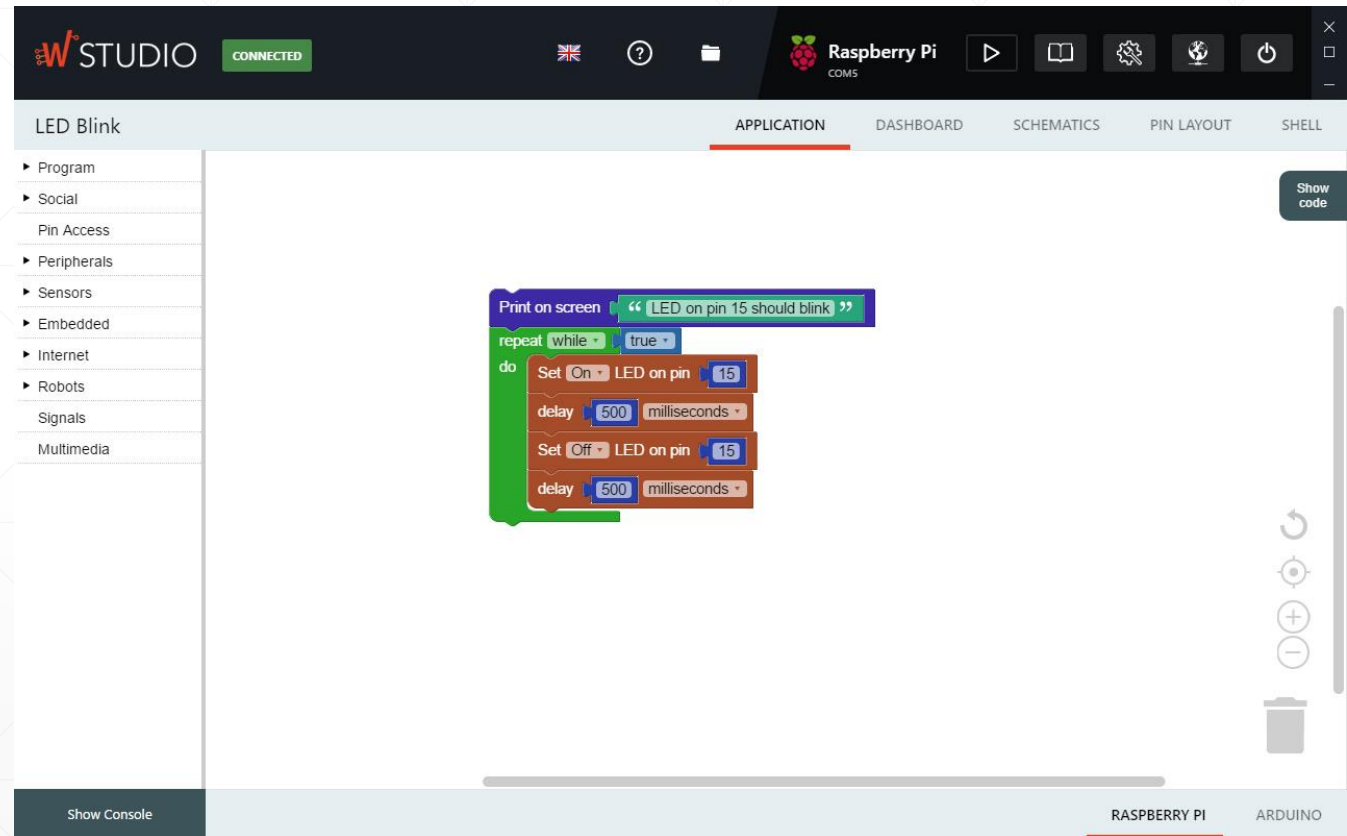
Implementation of
node-red



Visual Programming

Drag and drop blocks
of code

Implements Google
Blockly



View the source

View as Python
code gets generated

The screenshot displays the W Studio interface, which is connected to a Raspberry Pi. The interface is divided into several sections:

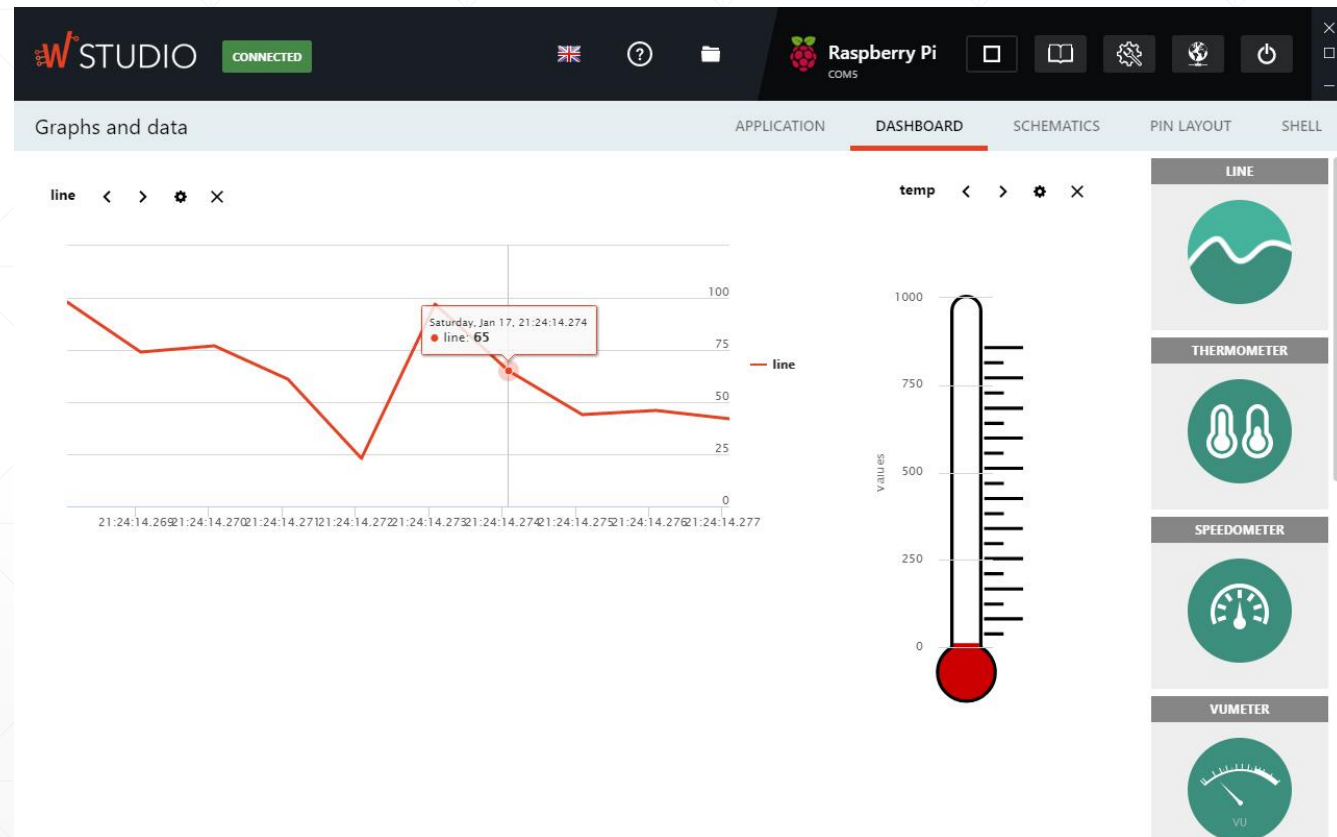
- Top Bar:** Includes the W Studio logo, a 'CONNECTED' status indicator, a language selector (set to English), a help icon, a file explorer icon, and a Raspberry Pi COM5 connection status.
- Left Sidebar:** A navigation menu for the 'Arduino sensor' project, listing categories like Program, Social, Pin Access, Peripherals, Sensors, Embedded, Internet, Robots, Signals, and Multimedia.
- Center Canvas:** Contains a block-based program. It starts with a 'Start Arduino on port "/dev/ttyUSB0"' block, followed by a 'Repeat every 1 seconds' loop. Inside the loop, there is a 'do' block containing a 'Send signal "light" with value' block, which is connected to an 'Analog read pin 0' block.
- Right Panel:** Shows the Python code generated from the block-based program. The code includes imports for pyfirmata, util, and threading, a function to set the board type, and a loop function that reads the analog pin and sends a signal.
- Bottom Bar:** Features a 'Show Console' button and a tabbed interface with 'RASPBERRY PI' and 'ARDUINO' tabs.

```
1 from pyfirmata import Arduino, ArduinoMega
2 from pyfirmata import util
3
4 from wylodrin import *
5
6 from threading import Timer
7
8 def setBoard(boardType, port):
9     if boardType == 'arduino':
10         board = Arduino(port)
11     else:
12         board = ArduinoMega(port)
13     return board
14 board = setBoard('arduino', '/dev/ttyUSB0')
15 reader = util.Iterator(board)
16 reader.start()
17
18 pin_var = board.get_pin("a:0:i")
19
20
21 def loopCode():
22     sendSignal('light', (round((pin_var.read() or 0)
23     Timer(1, loopCode).start()
24     loopCode()
25
```

Debug

Send signals to
dashboard

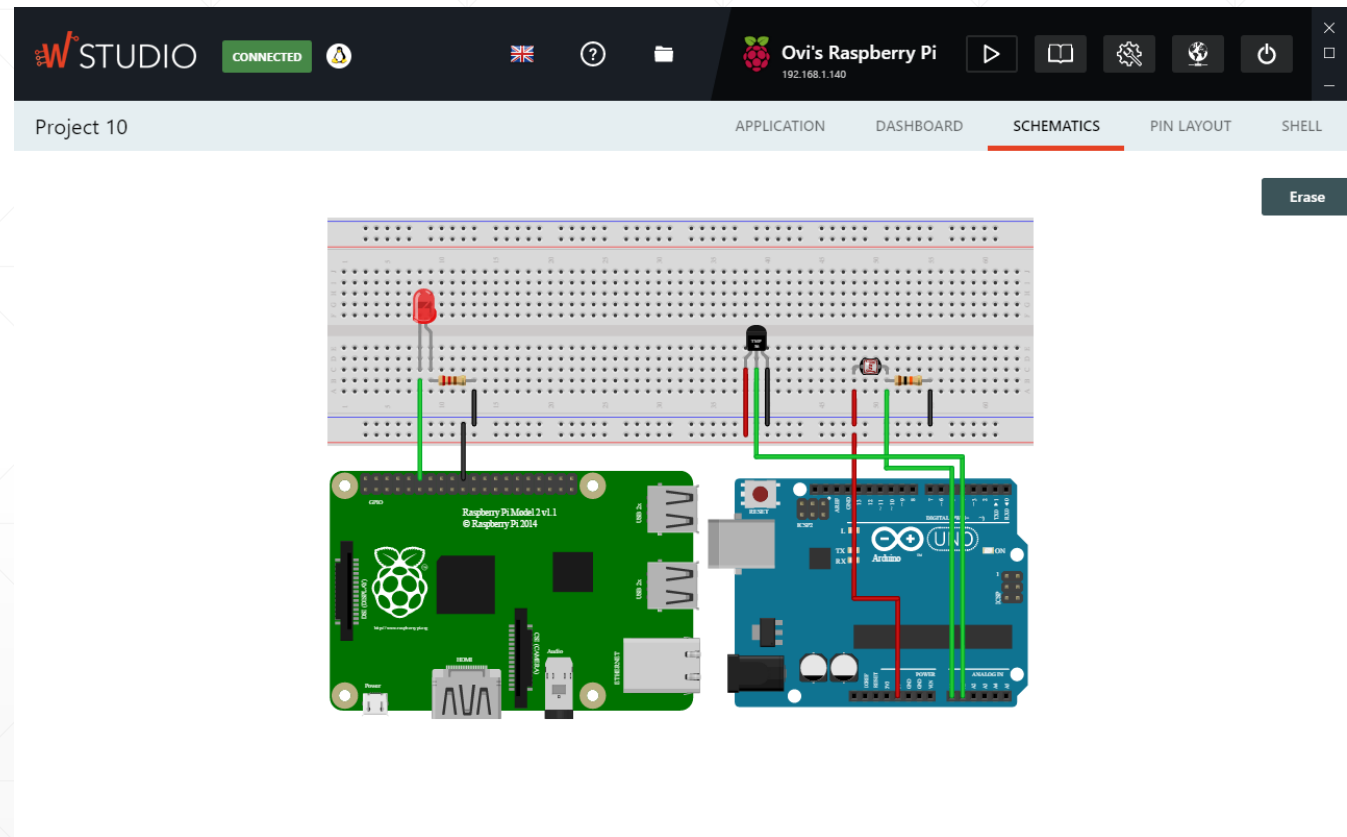
Put flags on graphs



Fritzing Schemas

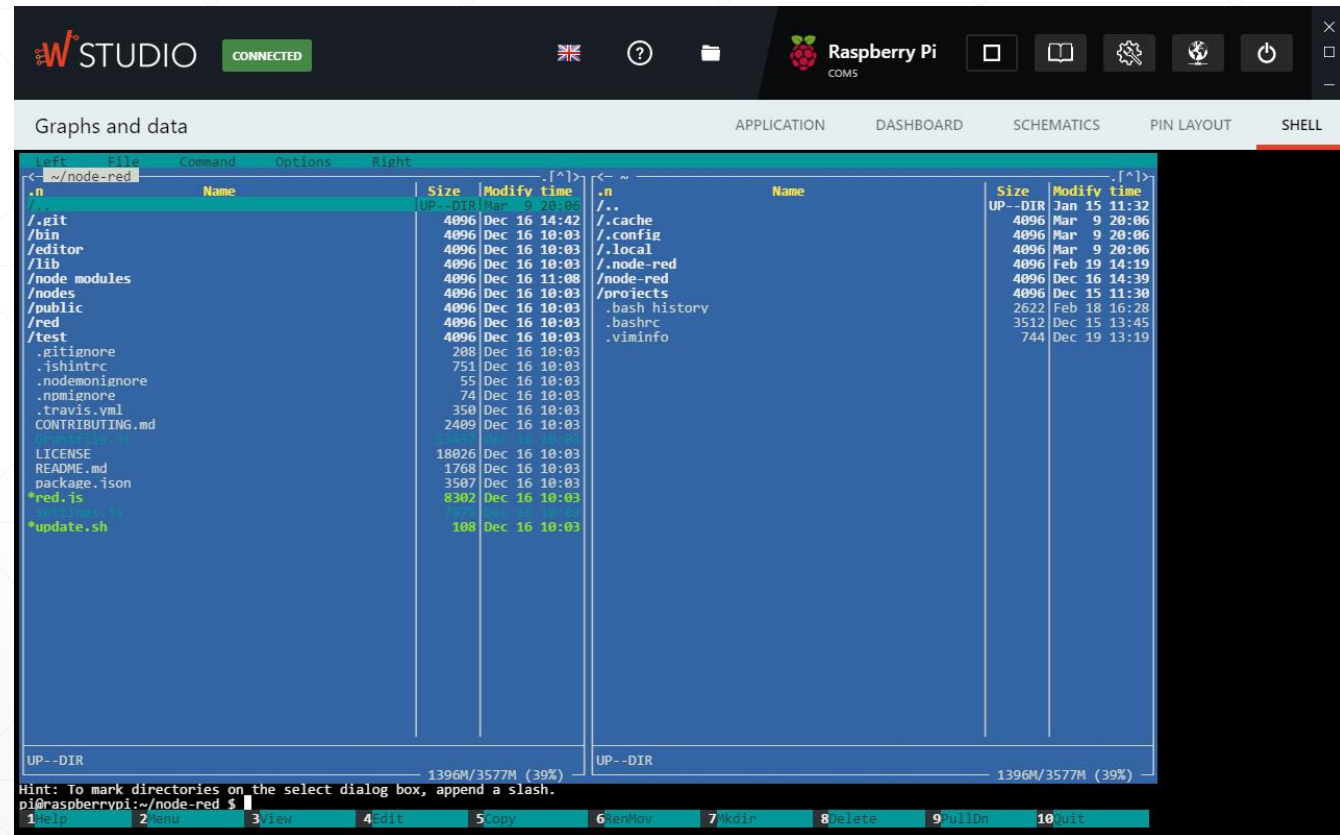
Import SVG from
Fritzing

Attach schema to
application



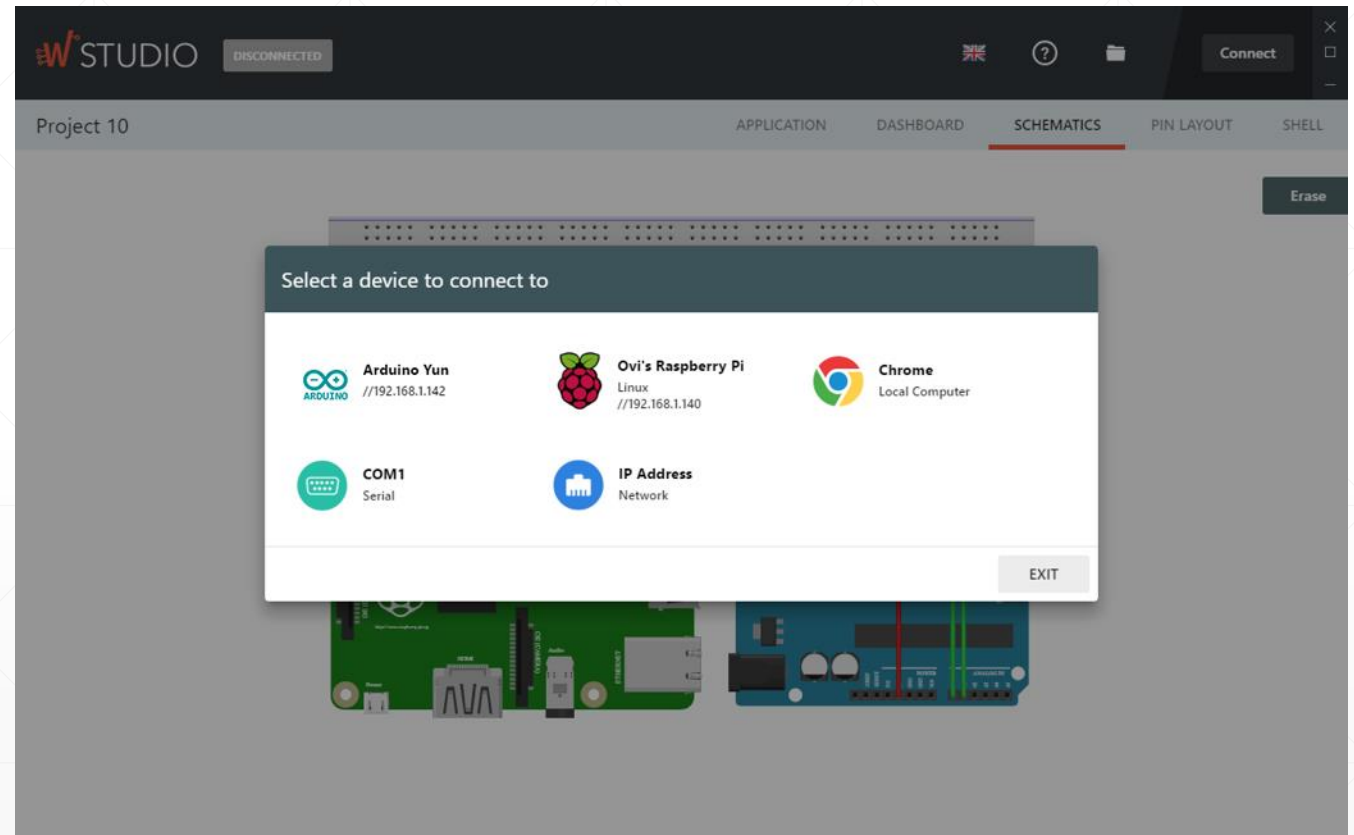
Shell

Direct shell for
advanced users



Board manager

- Visually Manages:
 - Network connections;
 - Libraries;
 - Tasks;
 - Projects.



libwyliodrin

- Open Source library
 - Universal API for pin control and board communication
 - Compatible with:
 - Arduino Yun
 - Raspberry Pi
 - Intel Galileo
 - Intel Edison
 - BeagleBone Black
 - UDOO Neo
-

Wylidorin STUDIO: future steps

- Enlarge the community
- Lessons
- Hardware simulation
- Projects sharing

Thank You!

Any questions?